REMARKS

The claims are claims 2 to 7 and 13 to 17.

The application has been amended at several locations to correct minor errors and to present uniform language throughout.

Claims 2, 7, 13 and 16 are amended. Claims 1, 8 to 12, 18 and 19 are canceled. Claims 2, 7 and 13 are amended to include all the limitations of prior base claim 1. Claim 16 is amended to delete limitations included in base claim 13.

An Information Disclosure Statement naming the patents cited on pages 7 and 11 of this application together with the required fee are attached.

Claims 1 to 19 were provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1 to 24 of copending Patent Application No. 09/998,756, claims 1 to 13 of copending Patent Application No. 09/998,755 and 1 to 10 of copending Patent Application No. 09/998,330.

A Terminal Disclaimer relative to U.S. Patent Application Serial Nos. 09/998,756, 09/998,755 and 09/998,330 is attached. This Terminal Disclaimer obviates any provisional obviousness-type double patenting rejection.

Claims 1 to 5, 7, 8 and 13 to 19 were rejected under 35 U.S.C. 103(a) as made obvious by the combination of Falik et al U.S. Patent No. 6,065,078 and Tarui et al U.S. Patent No. 6,088,770.

Claims 2, 7 and 13 recite subject matter not made obvious by the combination of Falik et al and Tarui et al. Claims 2, 7 and 13 each recite "creating a software memory map of how each of the plurality of processors may access memory including whether each of said plurality of processors may read from and write to a range of memory or may only read from said range of memory." The memory map taught in Tarui et al fails to indicate whether the processor may read/write or read only some parts of memory. Accordingly, claim 2 is not made obvious by the combination of Falik et al and Tarui et al.

Claim 2 recites further subject matter not made obvious by the combination of Falik et al and Tarui et al. Claim 2 recites the alternative actions of selecting the first or second processor to perform a write request depending on whether "the first processor associated with the first debug session has write access to the shared memory location." The OFFICE ACTION cites Tarui et al at column 9, lines 1 to 11 and column 8, lines 7 to 11 as making this limitation obvious. The cited portion of Tarui et al at column 99 begins "When a command has been delivered onto the shared bus. This shared bus is shared bus 120 illustrated in Figure 1A. As clearly shown in Figure 1A, shared bus 120 connects CPU's 110, 111 and 112. Thus a command received on the shared bus 120 must come from a CPU other than the CPU. Thus this is a command already passed from another processor and cannot be determining whether the first processor may service a write request. Tarui et al at column 8, lines 7 to 11 describes an Invalidate command that can be received via shared bus 120. This is not a write command but "a request to invalidate data retained in any other cache." Accordingly, claim 2 is not made obvious by the combination of Falik et al and Tarui et al.

Claim 2 recites further subject matter not made obvious by the combination of Falik et al and Tarui et al. Claim 2 recites "passing the write request initiated by the first debug session to the selected processor for execution." This write request is to "a shared memory location." The OFFICE ACTION cites Tarui et al at column 14, lines 22 to 26 and column 15, lines 46 to 52 as making obvious this subject matter. Both these passages refer to a "WB command." This WB command is defined at column 8, lines 12 to 15, which state:

"Write Back (WB)

"This command makes a request for the write back of a cache line. It arises when data in cache has been replaced."

It is known in the art that such "a request for the write back of a cache line" involves writing data stored in a cache to a corresponding main memory location. However, there is no requirement in this disclosure of Tarui et al that this main memory location is to shared memory. This portion of Tarui et al does not require the write back command to be to a memory location shared among plural processors. Accordingly, claim 2 is not made obvious by the combination of Falik et al and Tarui et al.

Claim 4 recites subject matter not made obvious by the combination of Falik et al and Tarui et al. Claim 4 recites "indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request." The OFFICIAL ACTION cites the I command of Tarui et al disclosed at column 14, line 5 to 14, and column 8, lines 7 to 11 as making obvious this subject matter. Tarui et al states at column 8, lines 7 to 11:

"Invalidate (I)
"This command makes a request to invalidate data retained in any other cache. It is issued in a case where the CPU has presented a write request to a cache line which is shared with any other cache. It is one of the CCC commands."

The Applicants submit that this invalidate command is not a write command. One skilled in the art would interpret the above quoted portion of Tarui et al as requiring that a portion of cache be marked invalid and no longer used to supply data. Claim 4 recites a maintaining cache coherency by writing the changed data to other processors' caches ("the write request is intended for maintaining cache coherency"). This differs from the manner taught in Tarui et al of invalidating cache entries. Accordingly, claim 4 is not made obvious by the combination of Falik et al and Tarui et al.

Claim 5 recites subject matter not made obvious by the combination of Falik et al and Tarui et al. Claim 5 recites "using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency." The OFFICE ACTION cites the invalidate command taught in Tarui et al at column 8, lines 7 to 11 as making obvious this subject matter. However, this invalidate command is not the write request recited in claim 5. Accordingly, claim 5 is not made obvious by the combination of Falik et al and Tarui et al.

Claim 7 recites further subject matter not made obvious by the combination of Falik et al and Tarui et al. Claim 7 recites "setting a first software breakpoint in a shared memory location in the first debug session such that all debug sessions are notified of the setting of the breakpoint." The OFFICE ACTION cites Falik et al at column 16, lines 40 to 50 as making obvious this subject matter. This portion of Falik et al states:

> "When the core reaches the breakpoint it will write to the DBGABORT register a "1" to the position of all processors 1840a to 1840c it needs to stop (usually all except itself). As a result, an ISE interrupt will be sent to all the processors whose respective bit in the ABORT register is set 1840a to 1840c and their program flow will be interrupted. Note that if the processor is already executing within monitor code, there will be no ISE interrupt and only the ABORT bit will be set. The monitor of each stopped processor will need to respond to the ISE interrupt the next time it polls on DBGISESRCA register."

The Applicants respectfully submit this portion of Falik et al fails to teach the recited "setting a first software breakpoint in a shared memory location." This recitation corresponds to the processes of establishing a breakpoint in the shared memory. Instead this teaches actions taken upon reaching a breakpoint ("When the core reaches the breakpoint"). Accordingly, claim 7 is not made obvious by the combination of Falik et al and Tarui et al.

Claim 7 recites still further subject matter not made obvious by the combination of Falik et al and Tarui et al. Claim 7 recites "clearing the first software breakpoint in the shared memory location in the second debug session such that all debug sessions are notified of the clearing of the breakpoint." The OFFICE ACTION cites Falik et al at column 17, lines 37 to 48 as making obvious this subject matter. This portion of Falik et al states:

> "When the first Rx message is received (identified by a polling loop on the Rx status register), the ABORT bit in DBGISESRCA register can be cleared by writing 1 to it. Operation of the application by the first processor 1840a is resumed when the first processor 1840a receives a "go" or "sgo" command from the debugger.
>
> "'go'=return from monitor to application. 'sgo'=all currently stopped monitors should start running together. Synchronization is achieved by meaans of shared memory or shared semaphore."

The Applicants respectfully submit this portion of Falik et al fails to teach the recited "clearing the first software breakpoint in the shared memory location." This recitation corresponds to the processes of removing a breakpoint in the shared memory. Removal of a breakpoint thus eliminates the possibility of a stopping processing at the location. Instead this teaches actions taken following reaching a breakpoint. Clearing the ABORT bit is part of the process of reacting to reaching a breakpoint. Accordingly, claim 7 is not made obvious by the combination of Falik et al and Tarui et al.

Claim 13 recites further subject matter not made obvious by the combination of Falik et al and Tarui et al. Claim 13 recites "denoting in the software memory map the shared memory locations whether or not each processor of the plurality of processors has a cache." The OFFICE ACTION cites Tarui et al at column 6, line 66 to column 7, line 9 and column 3, lines 5 to 11 as making obvious this limitation. Tarui et al states at column 6, line 66 of column 7, line 9:

> "A RAT (Remote Access Table) 138 is a circuit for storing therein the attributes (whether or not a page has been accessed

from any other node, and if the CCC to any other node is necessary or not) of each of the pages of the main memory included in the particular node. A RAT check circuit 133 is a circuit for checking the RAT value of the address accessed from the CPU and for starting a necessary operation. A RAT alternation circuit 148 is a circuit for altering the RAT value of the page accessed from any other node. These circuits 138, 133 and 148 are peculiar to this aspect of performance."

This portion of Tarui et al states that the RAT 138 stores data indicating whether a cache coherency check is needed. This portion of Tarui et al fails to state that the RAT 138 indicates whether a particular processor has cache as required by this limitation of claim 13. Tarui et at states at column 3, lines 5 to 11:

> "In actuality, however, the transaction for maintaining the cache coherency must be submitted to the other processors so as to check the caches of the other processors (hereinbelow, this processing shall be called the "CCC: Cache Coherent Check"). This is because there is a possibility that the copy of the accessed data has been buffered in the cache of another processor."

This portion of Tarui et al discloses a cache coherency check to maintain cache coherency when more than one cache can store the same data. This portion of Tarui et al fails to state that the cache coherency check indicates whether a particular processor has cache as required by this limitation of claim 13. Note further that Tarui et al states at column 6, lines 18 and 22:

> "More specifically, each of the nodes includes CPUs 110~112 including caches 110a~112a, a (partial) main memory 160, a main memory access circuitry 130, a network command transmission circuit 180, and a network command reception circuits 190."

Thus Tarui et al clearly states that all processors (CPUs 110~112) include corresponding cache. Thus Tarui et al would have no disclosure of denoting in the software memory map whether a particular processor includes cache. Accordingly, claim 13 is not made obvious by the combination of Falik et al and Tarui et al.

Claim 13 recites further subject matter not made obvious by the combination of Falik et al and Tarui et al. Claims 13 recites "searching the software representation of the memory map for a first plurality of processors that have read access to the shared memory location." The OFFICE ACTION cites Tarui et al at column 6, line 66 column 7, line 9 as making obvious this limitation. This portion of Tarui et al is quoted above. This portion of Tarui et al discloses that the RAT 138 stores data indicating whether a cache coherency check is needed. This portion of Tarui et al fails to state that the RAT 138 indicates which processors can access a particular memory address. Thus this determination cannot indicate whether processors have read access to a particular shared memory location as required by the limitation of claim 13. Accordingly, claim 13 is not made obvious by the combination of Falik et al and Tarui et al.

Claims 14 and 17 recite subject matter not made obvious by the combination of Falik et al and Tarui et al. Claims 14 and 17 recite "indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request." The OFFICE ACTION cites Tarui et al at column 14, lines 5 to 14 and column 8, lines 7 to 11 as making this limitation obvious. Tarui et al states at column 14, lines 5 to 14:

> "(B1) Case where CCCs toward Other nodes are Necessary
>
> "The RAT check circuit 133 requests the network command generation circuit 134 through the signal line 133a to generate the I command toward the other nodes (there is no reply to the I command). The network command generation circuit 134 sends the command to the nodes designated by the destination generation circuit 139. More specifically, the I command toward the local area is multicast to all the nodes within the partition, while the I command toward the shared area is broadcast to all the nodes of the system."

Tarui et al at column 8, lines 7 to 11 provides a definition of this invalidate command. The Applicants submit that this invalidate command is not the write request recited in claims 14 and 17. Distinguishing between a normal write and a write for cache coherency is not made obvious by the invalidate command taught in Tarui et al. Accordingly, claims 14 and 17 not made obvious by the combination of Falik et al and Tarui et al.

Claim 6 was rejected under 35 U.S.C. 103(a) as made obvious by the combination of Falik et al U.S. Patent No. 6,065,078, Tarui et al U.S. Patent No. 6,088,770 and Kahle et al U.S. Patent No. 6,539,500.

Claim 6 recites subject matter not made obvious by the combination of Falik et al, Tarui et al and Kahle et al. Claim 6 recites "denoting in the software memory map all the shared memory locations that contain program instructions upon each initialization of the target system." The OFFICE ACTION cites Kahle et al at column 3, lines 19 to 27 as making obvious this limitation. Kahle et al states at column 19, lines 19 to 27:

> "The processors 100 and 103 would need to have each instruction mapped to a trace register as the instruction is accessed for execution. As an instruction is being executed it can then be simultaneously written via bus 200 into the shared trace buffer. The hardware of processors 100 and 103 would also have registers 101 for maintaining pointers to the addresses in shared memory that define the space used for a particular trace of executed instructions."

This portion of Kahle et al specifically states the instruction are mapped to the trace register "the instruction is accessed for execution" and written into the shared trace buffer as "an instruction is being executed." The process as the instruction is being accessed and executed taught in Kahle et al occurs at a different time than the "upon each initialization of the target system." Further, this process disclosed in Kahle et al operates only upon instructions access and executed and not upon "all the shared memory locations that contain program instructions." Accordingly, claim 6 is not made obvious by the combination of Falik et al, Tarui et al and Kahle et al.

The Applicants respectfully submit that all the present claims are allowable for the reasons set forth above. Therefore early reconsideration and advance to issue are respectfully requested.

If the Examiner has any questions or other correspondence regarding this application, Applicants request that the Examiner contact Applicants' attorney at the below listed telephone number and address to facilitate prosecution.

Texas Instruments Incorporated
P.O. Box 655474 M/S 3999
Dallas, Texas 75265
(972) 917-5290
Fax: (972) 917-4418

Respectfully submitted,
Robert D Marshall
Robert D. Marshall, Jr.
Reg. No. 28,527